DISEÑO DE PROCESADORES DEDICADOS

Práctica 2
"Síntesis Lógica"
Tarjeta Spartan II

M. en C. Juan Carlos Herrera Lozada
jlozada@ipn.mx

Instituto Politécnico Nacional
Centro de Innovación y Desarrollo Tecnológico en Cómputo
CIDETEC

Marzo 2009

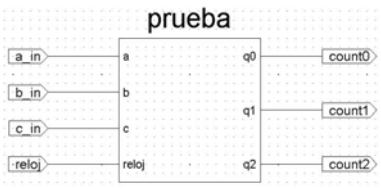
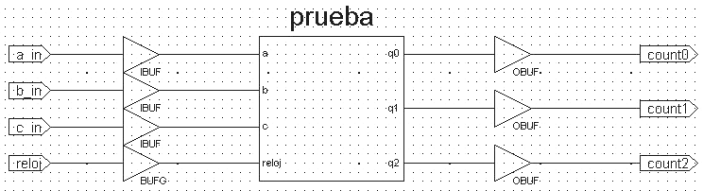
Campo 1: **Datos Personales.**

Campo 2: **Objetivos**.

- Síntesis Lógica y Programación de FPGA XC2S100.

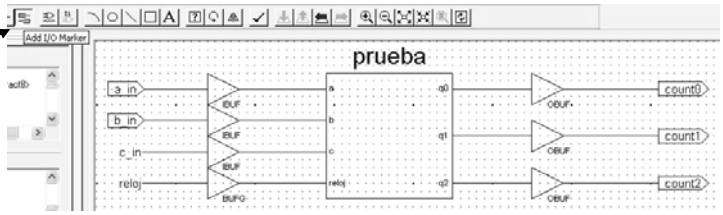
Campo 3: **Desarrollo de la Práctica**.

Antes de comenzar a utilizar la tarjeta, debes adecuar tus diagramas esquemáticos para la implementación física; esto es, independientemente del número de macros de tu esquemático, las variables en el nivel jerárquico más alto (entradas y salidas hacia el exterior) deben tener buffers y pines físicos (PADS). Dentro de una macro no puede haber PADS, sino símbolos a conectores de jerarquía baja. Lo anterior es importante debido a que el simulador no te indicará un error, pero las herramientas de síntesis sí lo harán. Observa la siguiente conversión.

Para simulación y/o para generar una macro	*Para implementación física*
	

Cada Pin debe incluir su respectivo buffer: *IBUF* es un buffer sólo para entradas y *OBUF* es un buffer sólo para salidas, ambos los obtienes de la biblioteca de *símbolos* dentro de la edición esquemática en la categoría *IO*.
BUFG es un buffer global que es exclusivo para señales de reloj y lo obtienes en la categoría *BUFFER*.
Si analizas las categorías indicadas podrás observar que es posible considerar buffers de entrada o de salida múltiples. De momento, sólo utilizaremos individuales.
Los PADs son los mismos *I/O Markers* que utilizas para las terminales en cualquier diagrama, disponibles en el menú principal de la edición esquemática. Es importante etiquetar las variables para que exista una referencia real al momento de asignar pines físicos.

I/O Markers, convencionales



El diseño con el que se comenzará a utilizar la tarjeta (DS-BD-2SLC SPARTAN II DEMO BOARD) es un contador hexadecimal que envía datos hacia uno de los displays a siete segmentos, incorporado en la misma tarjeta. Tienes que descargar el manual de usuario de la tarjeta y el código en VHDL, ambos materiales disponibles en línea (página del curso).

3.1 Generando el símbolo esquemático del contador de prueba

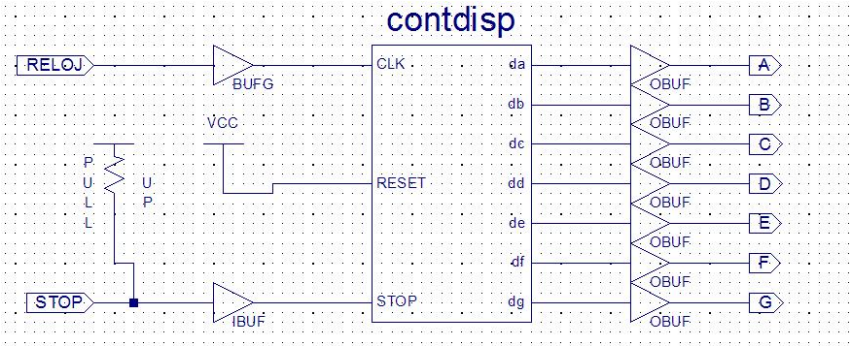
El código en VHDL listado a continuación denota el diseño propuesto. Para iniciar la práctica, revisa la sintaxis de este código y genera un símbolo esquemático con él. Consta de tres entradas: la señal de reloj, un reset y un stop. Como salidas del símbolo esquemático obtendrás los 7 segmentos del display de ánodo común (revisar el manual de la tarjeta).

```
--Maestría en Tecnología de Cómputo
--CIDETEC IPN, Diseño de Procesadores Dedicados
--Juan C. Herrera L. Prueba de tarjeta LC Spartan II.
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

ENTITY contdisp IS
  PORT(
         CLK, RESET, STOP: in STD_LOGIC;
         da, db, dc, dd, de, df, dg: out STD_LOGIC
           );
  END contdisp;

ARCHITECTURE archcontdisp OF contdisp IS
signal frec_contb: STD_LOGIC;
signal COUNT: STD_LOGIC_VECTOR (23 downto 0);
signal COUNTB: STD_LOGIC_VECTOR (3 downto 0);
signal display: STD_LOGIC_VECTOR (6 downto 0);
begin
divisor: process (CLK, RESET, COUNT)
begin
   if RESET='0' then
   COUNT <= "000000000000000000000000";
   elsif CLK='1' and CLK'event then
               COUNT <= COUNT + 1;
         else
               COUNT <= COUNT;
   end if;
       frec_contb <= COUNT(23);
end process divisor;

contabin: process (frec_contb, STOP)
begin
   if STOP='0' then
      COUNTB <= COUNTB;
   elsif frec_contb='1' and frec_contb'event then
               COUNTB <= COUNTB + 1;
         else
               COUNTB <= COUNTB;
   end if;
end process contabin;

decodisplay: process (COUNTB)
begin
--  CONTB(3:0) -> a|b|c|d|e|f|g , cátodo común
case COUNTB is
when "0000" => display <="1111110"; --0
when "0001" => display <="0110000"; --1
when "0010" => display <="1101101"; --2
when "0011" => display <="1111001"; --3
when "0100" => display <="0110011"; --4
when "0101" => display <="1011011"; --5
when "0110" => display <="1011111"; --6
when "0111" => display <="1110000"; --7
when "1000" => display <="1111111"; --8
when "1001" => display <="1110011"; --9
when "1010" => display <="1110111"; --A
when "1011" => display <="0011111"; --b
when "1100" => display <="1001110"; --C
when "1101" => display <="0111101"; --d
when "1110" => display <="1001111"; --E
when others => display <="1000111"; --F
end case;
end process decodisplay;

salidadisplay: process (display)
begin
da <= NOT display(6);
db <= NOT display(5);
```

```
dc <= NOT display(4);
dd <= NOT display(3);
de <= NOT display(2);
df <= NOT display(1);
dg <= NOT display(0);
end process salidadisplay;

END archcontdisp;
```

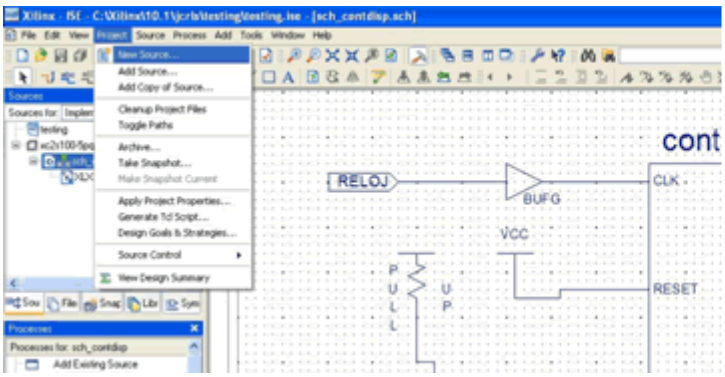
El circuito completo consta de la macro en VHDL con cuatro procesos: el primero (`divisor`) es el encargado de dividir el *reloj maestro de la tarjeta* de *25 MHz* a una frecuencia más baja que pueda ser utilizada en diseños de prueba. Este proceso de forma independiente y con modificaciones simples, te será muy útil para otros proyectos.
El segundo proceso, es propiamente el contador binario hexadecimal (`COUNTB`) que cambia con la frecuencia dividida del proceso anterior. El tercero de los procesos es un decodificador hexadecimal (`decodisplay`) a siete segmentos para un display de ánodo común, descrito de manera convencional.
El cuarto y último de los procesos (`salidadisplay`) invierte las salidas del decodificador hexadecimal para manejarlas individualmente y no en forma de bus, esta metodología de diseño será muy útil en la asignación de pines.

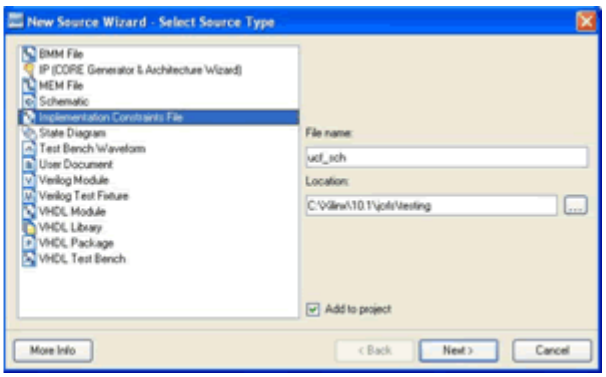
A continuación se muestra el diagrama esquemático que debes completar; para todas las líneas utiliza los mismos nombres que indica el diagrama esquemático. Observa que *RESET* está conectado a *VCC* con la intención de que siempre esté activo el contador que divide el tiempo. El componente *VCC* está disponible en la biblioteca de símbolos de la edición esquemática.
La señal de reloj se introducirá a través de un pad etiquetado como *RELOJ* al que se le conectará un *BUFG* dado que se trata de una señal variante en el tiempo y que se puede conectar a varios elementos a la vez.
De acuerdo a las características de la tarjeta (revisar manual), tanto los *push-buttons* como el *dipswitch*, no tienen resistencias alambradas, por lo que se requiere utilizar resistencias de *Pull-up* localizadas en la biblioteca de símbolos de la edición esquemática. Observa cómo se alambró la entrada *STOP* con su respectivo *IBUF*. Cada una de las salidas (para cada segmento del display) tiene su respectivo *OBUF*.



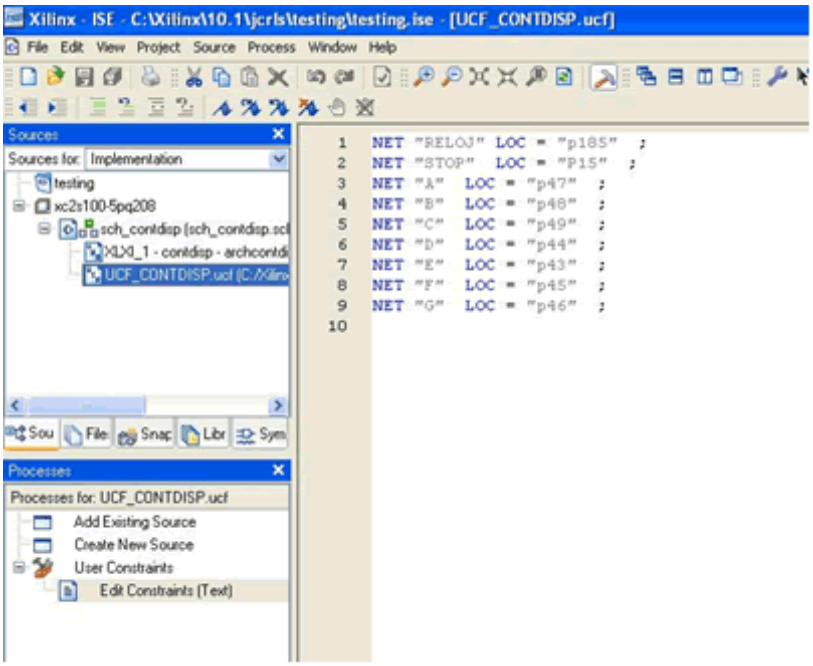
3.2 Asignando número a cada pin físico

Para asignar un número de pin, debes crear un archivo de pines *(UCF – User Constraint File)* que determinará la localización de cada pin desde un archivo de texto. Para ello, accede al menú principal y en *Project* selecciona *New Source*, posteriormente elige *Implementation Constraints File* y nombra tu archivo.





Notarás que el archivo *UCF* se adicionará automáticamente a tu proyecto. En *sources*, activa el archivo *UCF* con un clic del *Mouse*, accediendo a los procesos que podemos realizar sobre este archivo. Da doble clic sobre *Edit Constraints (Text)* y captura los datos indicados en la siguiente figura.


Xilinx - ISE - C:\Xilinx\10.1\jcrls\testing\testing.ise - [UCF_CONTDISP.ucf]
File Edit View Project Source Process Window Help
Sources
Sources for: Implementation
testing
xc2s100-5pq208
Processes
Processes for: UCF_CONTDISP.ucf
Add Existing Source
Create New Source
User Constraints
Edit Constraints (Text)
1 NET "RELOJ" LOC = "p185" ;
2 NET "STOP" LOC = "P15" ;
3 NET "A" LOC = "p47" ;
4 NET "B" LOC = "p48" ;
5 NET "C" LOC = "p49" ;
6 NET "D" LOC = "p44" ;
7 NET "E" LOC = "p43" ;
8 NET "F" LOC = "p45" ;
9 NET "G" LOC = "p46" ;
10

```
NET "RELOJ" LOC = "p185"  ;
NET "STOP"  LOC = "p15"  ;
NET "A"  LOC = "p47"  ;
NET "B"  LOC = "p48"  ;
NET "C"  LOC = "p49"  ;
NET "D"  LOC = "p44"  ;
NET "E"  LOC = "p43"  ;
NET "F"  LOC = "p45"  ;
NET "G"  LOC = "p46"  ;
```

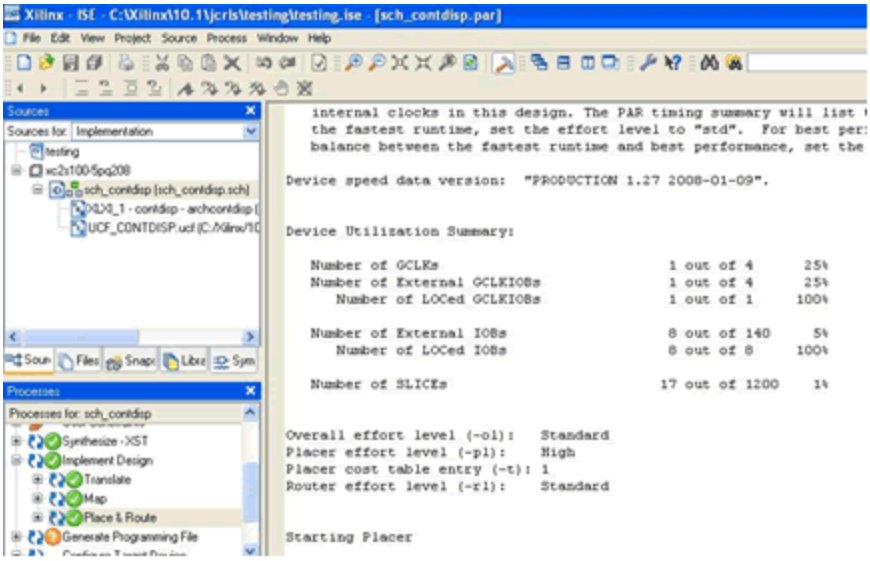
`LOC = p185`, indica que la variable *RELOJ* del diagrama esquemático está asignada al *pin 185* del FPGA en cuestión De igual forma, en el *p15* se tiene un push-button. La información de los pines "amarrados" para los displays y los que puedes utilizar con propósito general (push-buttons, dipswitch, leds), se consulta en el manual de usuario de la misma tarjeta. Es muy importante que el archivo *UCF* creado coincida con los nombres de las variables declaradas en los *I/O Markers (PADs)* de tu diagrama esquemático.

3.3 Implementación

Para la *implementación* de tu circuito (todas las fases del diseño hasta el archivo de bits que configurará al FPGA) debes estar en el *Project Navigator*, manipulando el archivo del diagrama esquemático que representa la jerarquía más alta de tu diseño y para el cual generaste un archivo *UCF* que determinará qué número de pin se le asignará a cada I/O Marker.

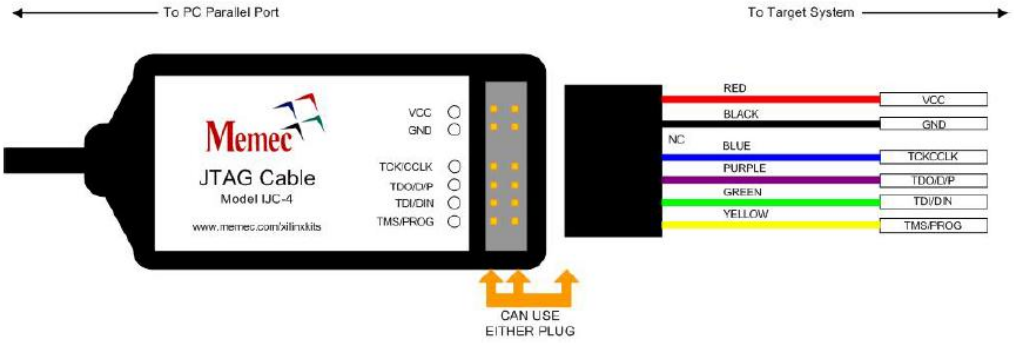
Comprueba que sólo esté el esquemático que deseas implementar (preferentemente); esto lo puedes verificar en la lista sensitiva de tu proyecto. Expande la opción *Implement Design* del área *Process for Source*, tal y como lo muestra la siguiente pantalla, a la izquierda. Posteriormente, trasládate a *Place & Route* y da doble clic sobre esta opción. Notarás que si todo es correcto, comenzarán a validarse todos los procesos de manera automática.

Elige el archivo *Place & Route Report* y observa la cantidad de recursos utilizados dentro del FPGA. Podrás advertir que el parámetro comparado es el número de *SLICEs* o *CLBs* (*Bloques Lógicos Configurables*) dentro del dispositivo. En ocasiones, al momento de realizar la implementación física obtienes errores derivados de conexiones flotantes (símbolos que aparecen juntos pero que no están conectados), pines definidos con el mismo nombre varias veces, pines que no tienen su respectivo buffer, o señales de reloj (señales de entrada que conectas a varios bloques) que no tienen un *BUFG* (buffer global) asignado. Abre el archivo *Pad Report* para verificar la asignación física de pines.

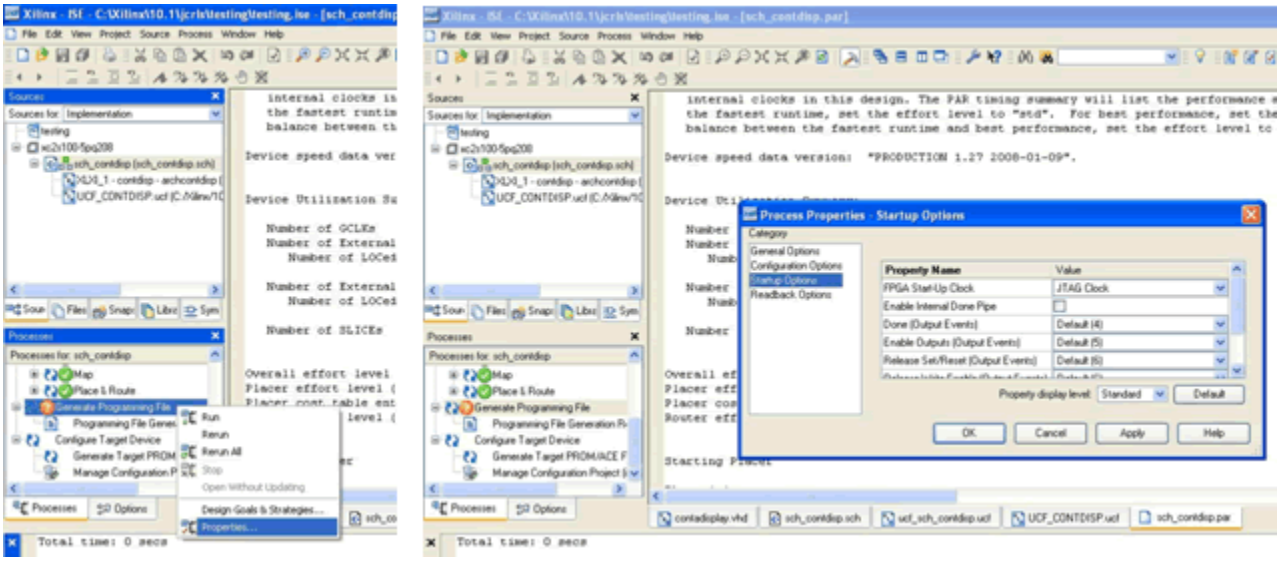

Xilinx - ISE - C:\Xilinx\10.1\jcrls\testing\testing.ise - [sch_contdisp.par]
File Edit View Project Source Process Window Help
Sources
Sources for: Implementation
testing
xc2v100-5pq208
sch_contdisp (sch_contdisp.sch)
UCF_CONTDISP.ucf
Processes
Processes for: sch_contdisp
Synthesize - XST
Implement Design
Translate
Map
Place & Route
Generate Programming File
internal clocks in this design. The PAR timing summary will list
the fastest runtime, set the effort level to "std". For best per
balance between the fastest runtime and best performance, set the
Device speed data version: "PRODUCTION 1.27 2008-01-09".
Device Utilization Summary:
Number of GCLKs 1 out of 4 25%
Number of External GCLKIOBs 1 out of 4 25%
Number of LOCed GCLKIOBs 1 out of 1 100%
Number of External IOBs 8 out of 140 5%
Number of LOCed IOBs 8 out of 8 100%
Number of SLICEs 17 out of 1200 1%
Overall effort level (-ol): Standard
Placer effort level (-pl): High
Placer cost table entry (-t): 1
Router effort level (-rl): Standard
Starting Placer

3.4 Programación

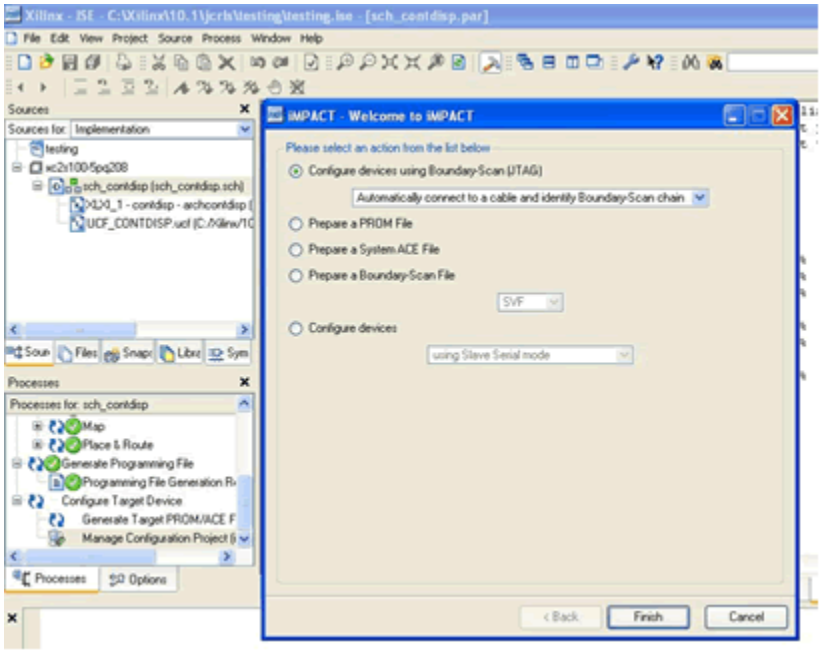
A partir de este momento debes tener conectado al puerto paralelo (LPT1) de la PC, el cable (JTAG Cable) y la tarjeta (alimentada, por supuesto). La tarjeta, el cable y los respectivos conectores están etiquetados para facilitar las conexiones entre ellos.



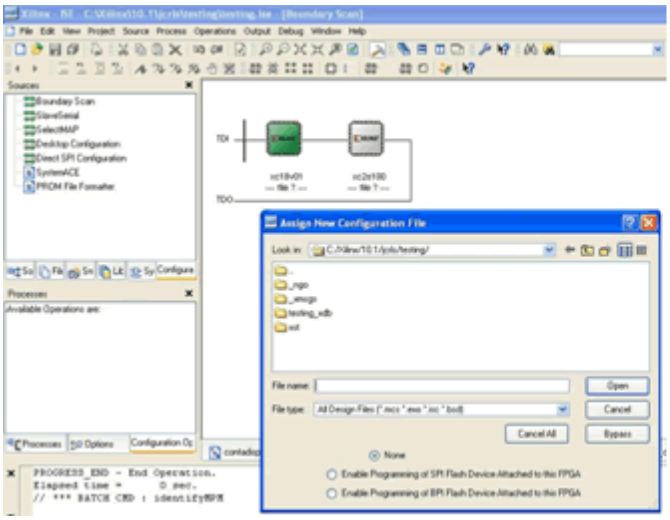
Posiciónate con un clic del *Mouse* sobre la opción *Generate Programming File* del área *Process for Source* y con el botón derecho del *mouse* accede a las propiedades de esta opción y dirígete a la pestaña *Startup Option*, y ahí, cambia la propiedad *Start-Up Clock* a un valor *JTAG Clock* (que es el tipo de cable que utilizaremos).

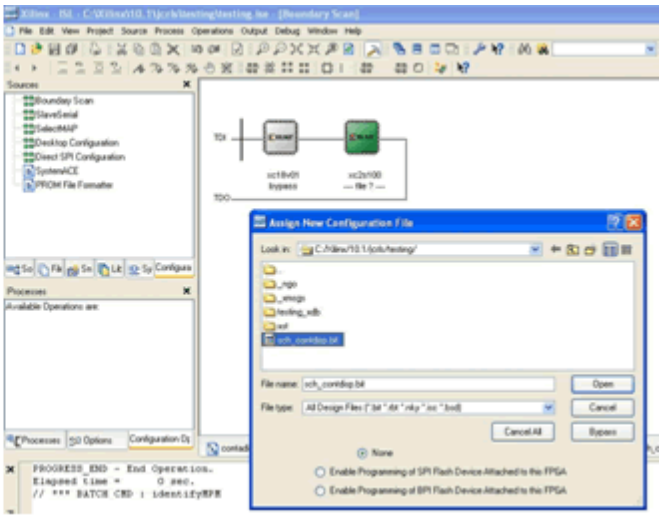


Al aceptar las modificaciones, regresa a desglosar los archivos de *Generate Programming File* y activa *Manage Configuration Project (iMPACT)*. Se te preguntará qué opción deseas para la interfaz con el *cable JTAG*; lo más común es mantener *Boundary – Scan*.

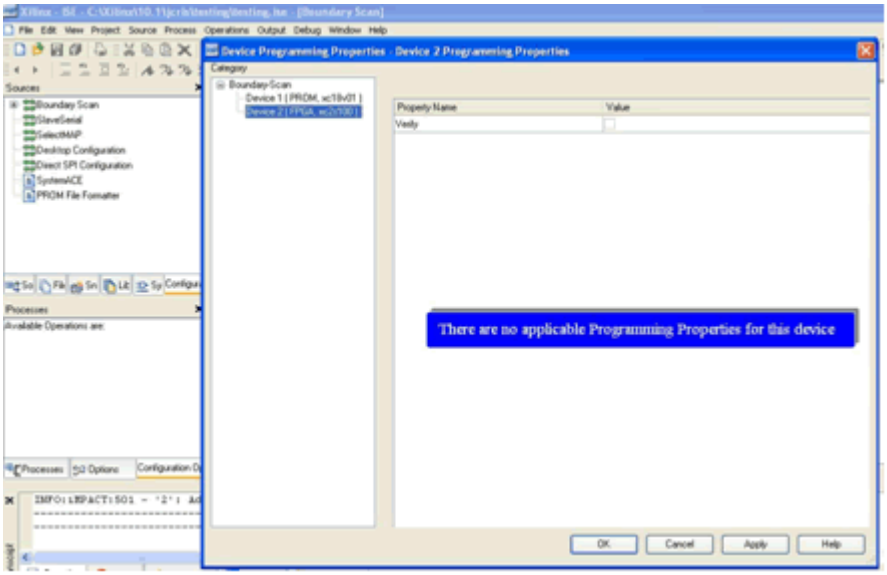

iMPACT - Welcome to iMPACT
Please select an action from the list below
Configure devices using Boundary-Scan (JTAG)
Automatically connect to a cable and identify Boundary-Scan chain
Prepare a PROM File
Prepare a System ACE File
Prepare a Boundary-Scan File
Configure devices
Finish
Cancel

La herramienta *iMPACT*, te mostrará una pantalla con dos chips, el de la izquierda es una memoria EEPROM (*XC18V01)* que contiene la tarjeta y que se puede programar para mantener una configuración; el chip de la derecha es el FPGA Spartan II XC2S100. Notarás que están conectados entre sí, conformando una cadena (*chain*) que comienza con la memoria EEPROM.
El proceso de esta cadena muestra una primera ventana (figura de abajo, a la izquierda) que te solicita un archivo relacionado con la memoria; ignórala cancelándola con la opción *Bypass* ya que nosotros no programaremos la memoria.
Posteriormente aparecerá un cuadro de diálogo que te solicitará el archivo de bits con el que se programará el FPGA (figura de abajo, a la derecha). Acepta el único archivo que aparece (debe ser el nombre de tu diagrama esquemático con la extensión *. *bit*).

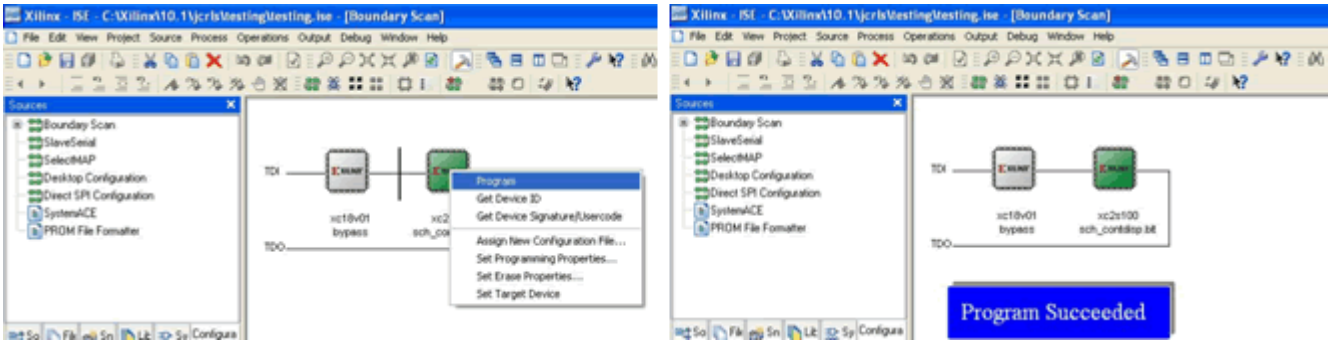




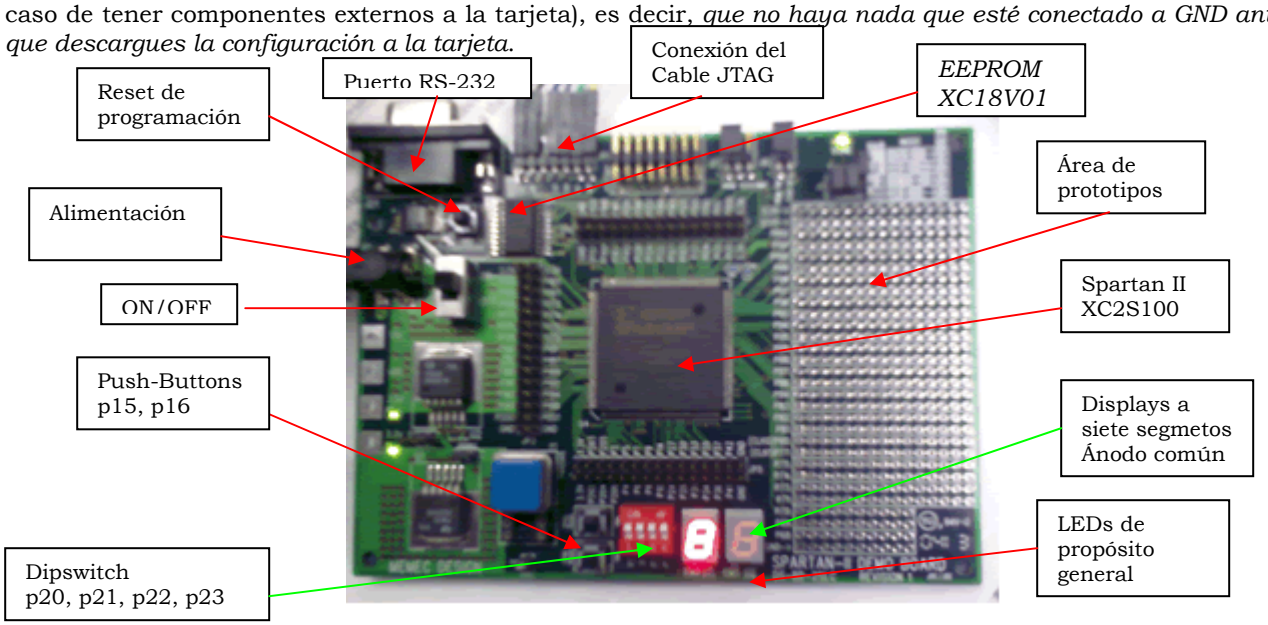
Una vez concluido este procedimiento, accederás a la pantalla de propiedades para la programación de ambos dispositivos (la EEPROM y el FPGA). No tenemos considerada alguna propiedad especial, por lo que sólo debes presionar *Cancel.*



Posiciónate sobre el FPGA (chip de la derecha en la ventana de iMPACT) y con el botón derecho del *Mouse* acciona *Program* para descargar tu archivo **.bit* al FPGA. También puedes hacerlo desde *Operations -> Program*. Aparecerán las opciones de la programación; mantenlas por omisión, presionando *OK*.



Está misma metodología es extensiva para todos tus proyectos. Es importante que recuerdes que el FPGA tiene algunos pines asignados para su configuración previa, que después cambian de función para acoplarse a tu diseño, por lo mismo es recomendable que desconectes momentáneamente GND de tu circuito armado (en el caso de tener componentes externos a la tarjeta), es decir, *que no haya nada que esté conectado a GND antes de que descargues la configuración a la tarjeta.*



Juan Carlos Herrera Lozada
jlozada@ipn.mx
CIDETEC IPN, México, 2009

Campo 4: **Validación de la Práctica.**

1. *(40 puntos)* Investiga el uso del IF-THEN-ELSE en alguna bibliografía de VHDL y modifica tu diseño para integrar un RESET a tu contador, activándolo a través del otro push-button de la tarjeta (pin 16). Recuerda auxiliarte en todo momento del manual de referencia de la tarjeta.

2. *(40 puntos)* Diseña un contador hexadecimal que con una variable de entrada externa (utiliza el pin 20, asignado a uno de los polos del dipswitch) permita cambiar el sentido del conteo para que sea ascendente-descendente. Mantén el push-button para el STOP en tu diseño.

3. *(20 puntos)* Realiza un listado sencillo de las principales características de la tarjeta Spartan II. Realiza una investigación sobre las familias de FPGAs de Xilinx e indica sus diferencias más importantes.

Campo 5: **Conclusiones individuales.**